VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015 Tomáš Bauer

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Bauer**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: E LINKX a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Gaura**

Konzultant bakalářské práce: Ing. Roman Hrdý

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Rád bych na tomto místě poděkoval všem, kteří mi během absolvování bakalářské praxe u této firmy pomáhali, především Ing. Romanu Hrdému a Ing. Radku Petříkovi, kteří mne doprovázeli během praxe a také Bc. Martinu Gürtlerovi za odborné rady.

Abstrakt

Tato bakalářská práce obsahuje popis a vypracování úkolů, které jsem dostával během odborné praxe ve firmě E LINKX a. s. Práce je rozdělená na dvě části. Obsahem první části práce je popis několika úkolů a dvou projektů, které jsem vypracoval během školení. V druhé části práce popisuji úkoly, které jsem vykonával v rámci firemních projektů. Tyto úkoly se týkaly testování webových aplikací, přes psaní testovacího scénáře, až po samotný vývoj nových modulů.

Klíčová slova: ASP.NET, C#, .NET, PowerShell, SQL, XML

Abstract

This bachelor thesis contains a description and working out tasks that I received during professional experience in company E LINKX a. s. Bachelor thesis is divided into two parts. The content of the first part is a description of several tasks and two projects that I worked out during training. The second part describes the tasks that I performed in the context of corporate projects. These tasks were related to testing of web applications, through the writing test scenarios, to the actual development of new modules.

Keywords: ASP.NET, C#, .NET, PowerShell, SQL, XML

Seznam použitých zkratek a symbolů

ASP	–	Active Server Pages
CRUD	–	Create, Read, Update, Delete
ERP	–	Enterprise resource planning
IIS	–	Internet Information Services
SQL	–	Structured Query Language
TFS	–	Team Foundation Server
T-SQL	–	Transact SQL
WCF	–	Windows Communication Foundation
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce popisuje průběh vykonávání odborné praxe u firmy ELINKX a. s. Tato firma mne velmi zaujala na přednášce firem, která se konala v areálu VŠB, kde jsem ji oslovil a po domluvě jsem byl pozván na pohovor. Na pohovoru jsem absolvoval znalostní test a poté jsem doma vypracoval zadaný úkol, který jsem následně odeslal. Po pár dnech jsem byl obeznámen s tím, že jsem přijat na praxi.

Práce je rozdělena do několika kapitol, kde každá kapitola má své vlastní vypracování. V kapitole 2 jsou základní informace o firmě a jejich vlastních produktech. V kapitole 3 jsou popsány nástroje a technologie, které jsem během praxe využíval k řešení zadaných problémů. V kapitole 4 jsou popsány pozice, na kterých jsem během praxe pracoval. V kapitole 5 je popis a vypracování projektů a úkolů, které jsem řešil během školení, buď samostatně, nebo v týmu. V kapitole 6 je popis a vypracování úkolů, které se již vztahovaly k firemním projektům. V kapitole 7 je popis využití znalostí získaných ze studia a popis znalostí chybějících během vykonávaní praxe. V kapitole 8 je závěrečné zhodnocení praxe.

2 O firmě

E LINKX a. s. je česká společnost, která působí na trhu již více než 10 let a je partnerem skupiny eD' system Group. Tato firma se zabývá vývojem a realizací podnikových ERP informačních systémů, které dokážou pokrýt ekonomické a evidenční funkce firem. Dále vyvíjí logistický systém, který řídí veškeré operace se zásilkou, a to od vzniku požadavku na přepravu zásilky, až po úplné ukončení procesu. E LINKX a. s. je členem Microsoft Partner Network a dlouholetým držitelem kompetence Gold Hosting [1].

2.1 xShop

xShop je chytrý e-shop splňující požadavky moderní doby s využitím responzivního designu, čili s možností pročítat stránky na jakémkoliv zařízení. Tento produkt již v základu obsahuje funkčnosti, které jsou typické pro ERP (objednávky, faktury, dodací listy, sklady). Pro náročnější je zde možnost zapnutí dalších modulů, které obsahují komplexní obchodní a účetní systém [2].

2.2 xSklad

xSklad je cloudové řešení pro řízení skladu. Uživatelé jednoduše přistupují k systému prostřednictvím internetového prohlížeče. Jádrem systému je warehouse management systém pro automatizované řízení skladových přesunů a optimalizaci skladových procesů [3].

2.3 Esyco.NET

Esyco.NET je ekonomický informační systém určen pro společnosti ze sféry obchodu a služeb. Systém je členěn do jednotlivých modulů, prostřednictvím kterých jsou potom prováděny jednotlivé procesy společnosti. Poskytuje moduly pro účetnictví, finance, skladové evidence, prodej, nákup, reklamace a logistiku [4].

3 Použité nástroje a technologie

V průběhu praxe jsem využíval různé nástroje a technologie, které jsem znal ze školy, ale také jsem si musel osvojit nové, které byly nezbytné pro vykonání praxe. Nejdůležitější a nejčastěji využívaný nástroj během praxe pro mne byl Visual Studio, ve kterém probíhal vývoj aplikací v jazyce C#, pro práci v týmu bylo nutností si osvojit práci s TFS. K práci s databází jsem využíval nástroj SQL Server Management Studio a v něm pomocí jazyka SQL, případně T-SQL, jsem tvořil jak tabulky, tak procedury pro práci s nimi, ale také často obtížnější dotazy pro získání požadovaných výsledků.

3.1 WinMerge

WinMerge je open source program, který slouží k porovnání a synchronizaci složek a souborů. Porovnávané soubory se vloží do panelu vedle sebe, a v případě rozdílnosti souboru provede zvýraznění řádků, ve kterých se rozdíly vyskytují. Díky tomu lze jednoduše dohledat rozdíly a případně upravit soubory, nebo je sloučit dohromady [6].

3.2 Team Foundation Server

TFS je placená aplikace od firmy Microsoft, která slouží převážně pro správu verzí nad daným projektem při práci v týmu. Díky tomu umožňuje členům týmu zapojit se do vývojového procesu pomocí jediného řešení. Dále poskytuje historii změn nad soubory, automatické řešení konfliktů, případně zobrazení konfliktů v souborech a poté je na uživateli, jak daný konflikt vyřeší. Důležitou funkcí je také možnost navrácení ke starší verzi v případě chyb v nové verzi [7].

3.3 SQL Server Management Studio

Je to nástroj od firmy Microsoft, který nám zajišťuje přístup k databázi pomocí jazyka SQL, případně T-SQL. Dále tento nástroj umožňuje provádět zálohování, nastavování přístupových práv, transakční zpracování a také replikace [8].

3.4 C#

C# je objektově orientovaný programovací jazyk firmy Microsoft, pomocí něhož lze psát různé typy aplikací, jako jsou databázové aplikace, webové aplikace, webové služby, formulářové aplikace, konzolové aplikace nebo služby pro Windows [9].

3.5 ASP.NET

ASP.NET je součástí .NET frameworku a využívá se pro tvorbu webových služeb a aplikací využívající HTML, CSS a JavaScript [10].

3.6 PowerShell

PowerShell je skriptovací jazyk od firmy Microsoft. Je založen na platformě .NET Framework, takže dokáže využívat všechny jeho knihovny. Obsahuje velké množství funkcí tzv. commandlets, což jsou speciální třídy pro určité operace [11].

3.7 HelpDesk

HelpDesk je produktem firmy E LINKX a. s. Firma tento produkt nabízí nejen na trhu, ale také jej sama využívá. HelpDesk slouží pro evidenci práce a požadavků. K danému požadavku lze přiřadit řešitele a přidat odhad (doba, za kterou by měl být daný požadavek hotov). Dále lze u požadavku měnit jeho stavy, připsat procentuální stav řešení a přidávat komentáře. Jakákoliv změna je okamžitě rozesílána e-mailem všem účastníkům na daném požadavku [5].

4 Pracovní zařazení

Během školení jsme byli rozděleni na týmy, ve kterých jsme pracovali na projektech a úkolech. Při práci na tvorbě transformačních souborů jsem byl na pozici vedoucího, kdy jsem měl za úkol koordinovat průběh práce na tomto úkolu a přidělovat dílčí části ostatním v týmu. V ostatních úkolech jsem již vedoucím nebyl, neboť jsme se ve vedení střídali, abychom si vyzkoušeli jaké to je být vedoucím.

Po skončení školení, kdy jsme byli přizváni, abychom pracovali na firemních projektech, jsem vystřídal hned několik zařazení. Nejprve jsem byl na pozici testera, kdy jsem měl za úkol testování aplikace, poté jsem psal testovací scénář pro moduly aplikace xforlink a nakonec jsem byl na pozici vývojáře, kde jsem vytvářel nové moduly a funkčnosti pro aplikace, které využívají přepravní společnosti.

5 Popis a vypracování úkolů a projektů během školení

Na začátku školení jsme dostali místnost, ve které jsme mohli pracovat na našich projektech a byli jsme rozděleni do týmů, ve kterých jsme dané projekty řešili. Dále jsme se domluvili, že se budou konat 1 až 2 krát týdně schůzky, na kterých budeme prezentovat naše dosavadní řešení za tým a případně se doptávat na nejasnosti v zadání či vyskytnuté problémy. Během práce na projektech jsme dostali i několik menších úkolů, které jsme řešili samostatně. Celkové školení trvalo něco málo přes dva měsíce.

5.1 Replikator

Prvním z projektů byla aplikace pro udržování synchronizace serverů a to tak, že změny provedené na hlavním počítači se musí replikovat na všechny ostatní počítače. Na aplikaci jsem pracoval s kolegou Michalem Libotovským přibližně jeden měsíc. Projekt jsme pojmenovali Replikator, protože toto řešení mělo primárně sloužit pro replikaci dat. Projekt jsme rozdělili do tří menších částí, které budu popisovat níže. Výsledkem projektu je fungující řešení s uživatelskou i technickou dokumentací.

5.1.1 Zadání

Zadání jsme dostali pouze slovní a na případné nejasnosti v zadání jsme se museli doptávat později. Níže je uvedeno ucelené slovní zadání.

5.1.1.1 Aplikace pro nastavení služby

Aplikace bude formou formuláře konfigurovat chod služby, která provádí synchronizaci. Ve formuláři bude možno přidat a odebrat adresy počítačů pomocí tlačítek a dále zde bude viditelný seznam s adresami všech počítačů, které se mají synchronizovat. Obdobně zde budou tlačítka pro přidání nebo odebrání hlídaných souborů. Dále zde bude možnost nastavení cesty k adresáři s historií, nastavení e-mailu a nastavení cesty k hlídanému adresáři. Nakonec zde budou prvky pro nastavení počtu verzí a nastavení zpoždění synchronizace v minutách. Při změně konfiguračního souboru se služba restartuje, aby pracovala s aktuálním nastavením.

5.1.1.2 Služba

Služba bude hlídat adresáře nebo soubory a při změně informuje uživatele, který potvrdí synchronizaci s ostatními počítači pouze v případě, že je spuštěna notifikační aplikace. Službu bude možno nastavovat přes aplikaci k tomu určenou. Služba po spuštění bude schopna se nakonfigurovat podle nastavení uloženého v konfiguračním souboru.

Při spuštění služby se provede kontrola, zda verze souborů na ostatních počítačích je stejná, a pokud ne, pošle se e-mail, informuje se uživatel a provede se synchronizace s aktuální verzí. Služba bude udržovat verze starých souborů v historii. V případě, že začne synchronizace a během ní se na hlavním počítači provede nová změna, další synchronizace se vloží do fronty a po skončení aktuální synchronizace se provede další v pořadí z fronty. Služba bude schopna pracovat bez notifikační aplikace. Veškeré události budou ukládány do logovacího systému ve Windows.

5.1.1.3 Notifikační aplikace

Aplikace bude fungovat tak, že když uživatel změní hlídaný soubor, tak vyskočí notifikace o změně s informací o souboru. Poté se zobrazí dialog, ve kterém uživatel rozhodne, zda se má synchronizace provést okamžitě, jestli se má odložit na později nebo se má stornovat a tím pádem dojde k vrácení změn v adresáři nebo souboru. V případě, že služba při spuštění zjistí jinou verzi souboru na jakémkoli vedlejším počítači, bude zobrazeno upozornění a následně bude odeslán e-mail s informacemi o tom, že někdo změnil soubor. Když při spuštění služba zjistí, že nemůže načíst z konfiguračního souboru data, tak se pomocí upozornění oznámí chyba uživateli a pošle se e-mail. Dále zde bude možnost spuštění aplikace pro nastavení služby a možnost pro připojení/odpojení notifikační aplikace od služby pro případ, že uživatel nechce být rušen notifikační aplikací. Počet instancí této aplikace není nijak omezen, tudíž je třeba řešit synchronizaci.

5.1.2 Řešení

5.1.2.1 Aplikace pro nastavení služby

Pro tuto aplikaci bylo potřeba vytvořit GUI a také způsob uložení konfiguračního nastavení služby do konfiguračního souboru. Dále bylo potřeba propojit aplikaci s konfiguračním souborem, aby se s ním dalo pracovat skrze aplikaci.

Prvním úkolem bylo vytvoření třídy pro sledované objekty. Tato třída obsahovala třídní proměnné, a to cestu ke sledovanému objektu, seznam cest, na které se bude replikovat, objekt třídy FileSystemWatcher pro sledování změn na dané cestě a unikátní pojmenování sledovaného objektu. Dále bylo potřeba přidat k této třídě dvě statické proměnné, a to seznam změn a seznam chyb. Poté bylo potřeba vytvořit metodu pro nastavení sledování objektu a odchytávání změn nad daným objektem.

Mým dalším úkolem bylo vytvoření knihovny pro práci s konfiguračním souborem. Nejprve jsem si napsal, jakou strukturu by měl konfigurační soubor mít. V konfiguračním souboru jsme potřebovali mít uloženo nastavení ve tvaru klíč:hodnota pro nastavení služby a poté sledované, kde bylo potřeba pro každý sledovaný mít napsáno, kam se bude daný sledovaný replikovat. Bylo tedy potřeba vytvořit v konfiguračním souboru

dvě sekce. Poté jsem si nastudoval, jak lze vytvářet vlastní sekce v konfiguračním souboru a následně jsem začal programovat.

Pro každou sekci bylo nutné vytvořit tři třídy, které dědily z tříd určených pro vytváření vlastních sekcí v konfiguračním souboru. První třída dědila z třídy ConfigurationElement. V této třídě jsem vytvořil dvě třídní proměnné, a to Name a Value. Tato třída měla za úkol držet údaj typu klíč:hodnota. Druhá třída dědila z třídy ConfigurationElementColletion. Ve třídě jsem přepsal metody pro vytvoření nového elementu a získání elementu typu ConfigurationElement. Tato třída sloužila pro vytvoření kolekce typu ConfigurationElement. Třetí třída dědila z třídy ConfigurationSection. Ve třídě jsem vytvořil třídní proměnnou typu ConfigurationElementColletion, která uchovávala kolekci. Obdobným způsobem jsem vypracoval i třídy pro sledované objekty. Dále jsem vytvořil dvě třídy, které měly tyto sekce reprezentovat v aplikaci. Nakonec jsem vytvořil třídu ConfigurationManipulation, ve které jsem vytvořil metody pro vytvoření požadovaných sekcí, načtení dat z konfiguračního souboru a uložení dat do konfiguračního souboru. Když jsem byl hotov a knihovna byla funkční a otestována, napojil jsem tuto knihovnu k aplikaci, a tam jsem použil metodu SaveToConfig pro uložení dat z aplikace a metodu LoadFromConfig pro načtení dat do aplikace a následné napojení na komponenty v GUI.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        <section name="otherSetting" type="ConfigurationAccessLibrary.OtherSettingSection,␣
            Version=1.0.0.0,␣ConfigurationAccessLibrary,␣Culture=neutral,PublicKeyToken=null"/>
        <section name="paths" type="ConfigurationAccessLibrary.PathSection,␣Version=1.0.0.0,
            ConfigurationAccessLibrary,␣␣Culture=neutral,␣PublicKeyToken=null"/>
    </configSections>
    <otherSetting>
        <setting>
            <otherSetting name="numOfHis" value="10" />
            <otherSetting name="delay" value="1" />
            <otherSetting name="email" value="bauer.tomas@centrum.cz" />
            <otherSetting name="pathToHistory" value="C:\Users\Historie" />
            <otherSetting name="smtpServerAdress" value="smtp.centrum.cz" />
            <otherSetting name="smtpPort" value="25" />
        </setting>
    </otherSetting>
    <paths>
        <watchedFiles>
            <File path="C:\Users\Tomas\Desktop\pokus\Dokument.txt" name="Doc">
                <pc>
                    <PC path="C:\Users\Tomas\Desktop\pokus1\" />
                </pc>
            </File>
        </watchedFiles>
    </paths>
</configuration>
```

Výpis 1: Výsledná struktura konfiguračního souboru

5.1.2.2 Služba

Pro tuto aplikaci jsem vytvořil třídu obsahující dva časovače a poté metody, které nastavovaly časovače podle toho, zda se jednalo o okamžitou replikaci, odloženou replikaci nebo zrušení replikace. Dále jsem implementoval metodu OnTime, která procházela frontu změn a poté se podle změny prováděla replikace. Během provádění replikace se vytvářely zprávy, které se následně posílaly na notifikační aplikaci. Notifikační aplikace tyto zprávy následně zobrazila uživateli. Další metodou byla metoda UndoOnTime, která měla opačný úkol oproti metodě OnTime a to ten, že místo toho, aby prováděla replikaci, měla za úkol replikaci dat zrušit. Nakonec jsem v této třídě vytvořil nutné metody pro vytváření a mazání složek a metody pro kopírování a mazání souborů. Při vytváření metody pro kopírování souboru na cílové cesty jsem přišel na dobrou optimalizaci, díky které jsem nemusel pokaždé provádět kopírování souboru, což by bylo výkonnostně náročné. Optimalizace spočívala v tom, že jsem si kopírovaný soubor jednou načetl do paměti za využití třídy MemoryStream a poté jsem jej jen ukládal na příslušné cesty.

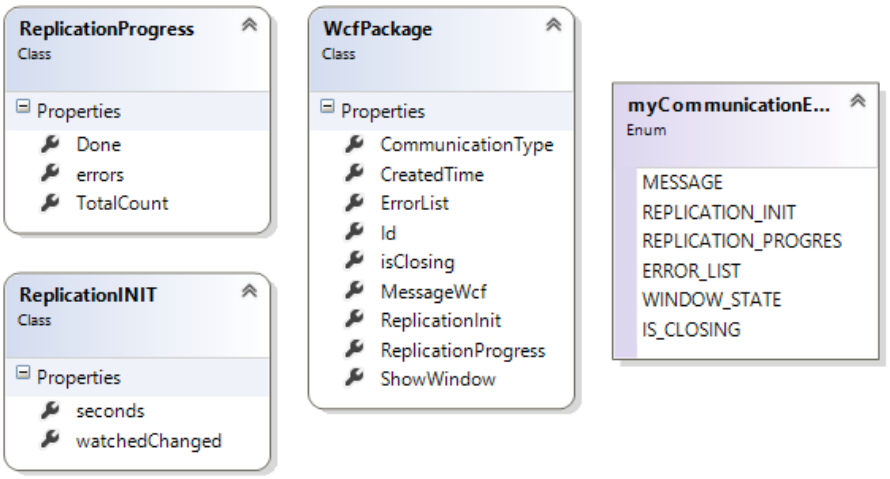
```
private void FileCreate(string fName)
{
    byte[] file = File.ReadAllBytes(invokedOn.FullName);
    using (MemoryStream memory = new MemoryStream(file))
    {
        foreach (string destName in commandOn.GetDestinations())
        {
            string path = null;
            string directoryPath = null;
            try
            {
                path = Path.Combine(destName, fName);
                directoryPath = path.Substring(0, path.Length - Path.GetFileName(path).Length);

                if (!Directory.Exists(directoryPath))
                    Directory.CreateDirectory(directoryPath);

                using (FileStream fs = new FileStream(path, FileMode.Create))
                {
                    byte[] data = memory.ToArray();
                    fs.Write(data, 0, data.Length);
                }
            }
        }
    }
}
```

Výpis 2: Metoda pro kopírování souboru na cílové cesty

5.1.2.3 Notifikační aplikace

Mým prvním úkolem bylo vytvoření okna, ve kterém se zobrazovaly případné chyby během provádění replikace. Tato úloha byla poměrně jednoduchá, protože okno obsahovalo pouze tabulku naplněnou chybami po skončení replikace.

Dalším úkolem bylo navázat a dokončit komunikaci mezi službou a notifikační aplikací. Již jsem dostal vytvořenou třídu WcfPackage, která obsahovala důležité informace pro posílání zpráv skrze WCF. Služba nic nevěděla o notifikační aplikaci, ani o tom, kolik těchto aplikací se služby dotazuje. Služba pouze vystavovala svoje rozhraní, které obsahovalo několik metod. Metoda Abort sloužila k tomu, aby bylo možné pomocí aplikace zrušit replikaci. Metoda AreYouHere sloužila pro zjištění, zda služba je spuštěná. Metoda DoNow prováděla okamžitou replikaci dat. Další metoda WaitThenDo prováděla odloženou replikaci. Nakonec nejzajímavější metoda DoYouHaveSomething vracela instanci WcfPackage, která obsahovala data pro notifikační aplikaci. V této metodě jsem plnil WcfPackage podle toho, které události nastaly. Tyto změny jsem jednoduše vkládal do instance WcfPackage, a to tak, že pokud daný objekt byl null, znamenalo to, že daná událost nenastala. Tímto postupem jsem naplnil balíček. Nakonec bylo nutné přiřadit mu ID. Toto ID bylo nutné z důvodu, aby notifikační aplikace nečetla stejná data dvakrát. Tímto řešením jsem obešel možnost, že aplikace, která čte data pravidelně v intervalu 1s nemůže přečíst stejná data dvakrát. Životnost balíčku byla 1s, což znamená, že všichni, kteří by volali metodu DoYouHaveSomething v rámci 1s získají shodná data. Po skončení platnosti balíčku se jednoduše vytvořil nový balíček.



Obrázek 1: Diagram tříd potřebných pro WCF komunikaci mezi aplikacemi

5.2 Webová aplikace pro nastavení desktopové aplikace

Druhým týmovým projektem bylo vytvoření webové aplikace v ASP.NET pro nastavování konfiguračního souboru replikační aplikace. Z důvodu, že webová aplikace má být spuštěná na jiném serveru než replikační aplikace, bylo nutné vytvořit i webovou službu, která vystavila metody, pomocí nichž webová aplikace pracovala s konfiguračním souborem.

Nejprve jsem začat tvořit webovou službu, ve které jsem implementoval metody, které jsou nutné pro čtení a zápis do konfiguračního souboru replikační aplikace. Pro získání nastavení služby, což byly klíč:hodnota atributy jsem vytvořil metodu, která vracela objekt, ve kterém byly tyto atributy zabaleny. Díky tomu se snížil počet dotazů na službu, protože nebylo nutné se dotazovat pro každý atribut zvlášť. Naopak pro ukládání bylo vhodné vytvořit pro každý atribut vlastní metodu, neboť při změně jednoho atributu není nutné ukládat i nezměněné. Ostatní metody byly implementovány dalšími členy týmu.

Ve druhé fázi jsem založil nový projekt pro webovou aplikaci a následně jsem vytvořil k aplikaci účty, což umožnilo mít veřejnou část aplikace a neveřejnou část aplikace. Dále jsem vytvořil lokalizaci aplikace pro český a anglický jazyk. Pro lokalizaci jsem použil nástroj zabudovaný ve Visual Studiu, který mi pro danou stránku vygeneroval soubor, ve kterém byly všechny textové zdroje dané stránky. Vygenerovaný soubor pro stránku O nás měl název About.aspx.resx. Abych mohl vytvořit lokalizaci pro anglický jazyk, bylo nutné vygenerovaný soubor zkopírovat a uložit jej pod názvem About.aspx.en-US.resx. Nakonec stačilo tento soubor otevřít a české texty nahradit texty anglickými. Aby bylo možné překlad použít v aplikaci, bylo nutné přidat do stránky přepínač, který po kliknutí změnil jazyk.

5.3 Tvorba transformačních XML souborů

Transformační XML soubor je soubor, který obsahuje rozdíly mezi dvěma konfiguračními soubory. Tento soubor obsahuje také speciální syntaxi, která je určená výhradně pro transformaci viz. zdrojový kód 3. V tomto úkolu šlo o vytvoření transformačních souborů pro konfigurační soubory ASP.NET aplikací.

Dostal jsme dva druhy konfiguračních souborů a to jedny staré a jedny nové. Poté jsem si vždycky otevřel ve WinMerge aplikaci jeden starý konfigurační soubor a jeden nový abych okamžitě viděl rozdíly a mohl jednoduše začít vytvářet transformační soubor. Jelikož mnoho hodnot atributů se shodovalo v rámci různých transformačních souborů, měl jsem za úkol tyto shody zapisovat jako konstanty do skriptu v powershellu pro jednodušší a rychlejší změny. Poté, když už jsem měl transformační soubor hotový, pustil jsem program, který provedl nahrazení konstant v transformačních souborech za hodnoty a samotnou transformaci. Nakonec jsem znovu otevřel dané soubory pro porovnání, zda

už mezi nimi není rozdíl. Pokud byl, pak jsem opět poupravil transformační soubor, jinak jsem pokračoval v tvorbě další transformace.

```
<?xml version="1.0"?>
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <xdt:Import path="C:\XDT\XDTComments.dll" namespace="XDTComments"/>
  <InsertContentBefore xml:space="preserve" xdt:Transform="InsertContentBefore(*)">
    <!-- RTM_eps.ppl.cz -->
  </InsertContentBefore>
  <connectionStrings xdt:Transform="RemoveCommentsRecursive(0,1)">
    <add name="EPS_NET" connectionString="{eps.ppl.cz-EPS_NET}" xdt:Transform="
        SetAttributes" xdt:Locator="Match(name)"/>
    <add name="EPS_NET_APPSERVER" connectionString="{eps.ppl.cz-
        EPS_NET_APPSERVER}" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>
  </connectionStrings>
  <appSettings>
    <add key="LogFileName" value="{eps.ppl.cz-LogFileName}" xdt:Transform="SetAttributes"
        xdt:Locator="Match(key)"/>
    <add key="TaskBoxWsUrl" value="{eps.ppl.cz-TaskBoxWsUrl}" xdt:Transform="SetAttributes"
        xdt:Locator="Match(key)"/>
  </appSettings>
  <system.web>
    <machineKey decryptionKey="{eps.ppl.cz-machinekey-DesK}" validationKey="{eps.ppl.cz-
        machinekey-ValK}" xdt:Transform="Insert"/>
  </system.web>
  <system.serviceModel>
    <client>
      <endpoint address="{eps.ppl.cz-Endpoint}" xdt:Transform="SetAttributes">
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

Výpis 3: Ukázka transformačního XML souboru

5.4 Vypracování prezentace o Feedech v souvislosti s porovnávači cen

Tento úkol byl první úkol z řady teoretických úkolů. Zadáním tohoto úkolu bylo zjistit co je to Feed a jak s ním porovnávače cen pracují a následně vytvořit prezentaci. U tohoto úkolu si mohli zadavatelé ověřit, jak dokážeme vyhledávat informace o něčem neznámém a následně to prezentovat a popsat ostatním.

Nejdříve jsem si našel, co to Feed znamená, abych mohl zjistit, jak vlastně funguje. Poté jsem si našel na stránkách porovnávače cen všechny důležité informace o tom, jak se může zboží dostat na stránky porovnávače a co je k tomu potřeba. Z těchto informací jsem poté vytvořil prezentaci a následně ji prezentoval ostatním.

5.5 Vypracováni analýzy, proč daná aplikace funguje pomalu

Druhý úkol z řady teoretických úkolů byl pojat z hlediska analýzy, aby si o nás mohli udělat představu, jak umíme splnit takové úkoly. Všichni jsme dostali přístup k webové aplikaci xforlink (aplikace, která se skládá z různých modulů, a to od správy e-shopu až po správu skladu) a shodné zadání obsahující otázky, na které jsme měli odpovědět ve své prezentaci, jež jsme měli vytvořit a následně prezentovat. V tomto úkolu jsem měl zodpovědět tyto otázky: Jaké jsou možné důvody, proč je aplikace u zákazníka pomalá? Jaký je nejlepší způsob analýzy chování aplikace přímo na počítači klienta?

Nejprve jsem zkoušel různá připojení k internetu, abych vyloučil možnost, že zákazník má pomalé připojení k internetu. Jelikož aplikace měla pomalou odezvu jak ve firmě, tak i u mě doma, bylo jasné, že připojením k internetu to není. Poté jsem našel nástroj, pomocí něhož jsem chod aplikace analyzoval. Použil jsem Developer tools z prohlížeče od společnosti Google, který dokáže graficky a v čase znázornit, co se při požadavcích uživatele (např. kliknutí na tlačítko) děje, a hlavně jaká data se stahují a jakou mají velikost a dobu stahování. Po analýze jsem vytvořil prezentaci a následně ji za přítomnosti Ing. Romana Hrdého a Ing. Radka Petříka prezentoval.

5.6 Tvorba nasazovacího skriptu pro automatizaci nasazení webového informačního systému a eshopu

Mým úkolem bylo rozšířit skript v PowerShellu pro automatizované nasazení webové aplikace do IIS, oddělit definice konstant od metod pro jednodušší a přehlednější úpravu a vytvořit logování událostí do souboru. K dispozici jsem dostal skript, který již obsahoval metody pro vytvoření nové Site, což je vlastně kontejner fyzických a virtuálních složek, které mají unikátní binding ve formátu IP:Port:HostHeader pro přístup k Site z internetu a metody pro vytvoření WebFarm, což je více serverů, na kterých běží stejná aplikace, a v případě vytíženosti serveru se ostatní požadavky posílají na ostatní servery, které jsou ve WebFarm pro danou aplikaci.

5.6.1 Přidání / odebrání UrlRewrite pravidel

UrlRewrite pravidlo umožňuje administrátorovi vytvořit pravidlo pro Url, které je jednoduché pro uživatele k zapamatování. Toto pravidlo také slouží k jednoduššímu vyhledání pomocí vyhledávacích nástrojů.

Prvním úkolem bylo vytvoření funkce se vstupními parametry wAppName pro název aplikace, pro kterou má být pravidlo vytvořeno a Url, které má být vloženo do pravidel. Po vytvoření hlavičky funkce jsem musel ošetřit vstupy funkce, a to tak, že pokud nějaký parametr nebyl zadán, byla vyhozená výjimka a ta byla následně vypsána do logu i konzole. Poté bylo nutné vytvořit funkce pro zjištění, zda Site a WebFarm pro danou aplikaci existuje a zda již taková podmínka v UrlRewrite pravidlech neexistuje. Jakmile

jsem měl všechny možné selhání vložení pravidla ošetřeny, tak zbývalo pouze vytvořit příkaz, který do konfiguračního souboru dané pravidlo uložil. Když už byla funkce hotová, začal jsem vytvářet funkci pro odebrání pravidla, která byla z velké části podobná té pro přidání.

5.6.2 Vložení / odstranění kontejneru z / do údržby

Kontejnerem se rozumí více Site, které mají část jména shodnou. Např. xFM_M0001 a xWS_M0001. Druhým úkolem bylo vložení kontejneru do údržby, což znamenalo vzít UrlRewrite pravidla ze všech Sites které měly jít do údržby a vložit je k Site, která obsahovala statickou stránku s upozorněním, že probíhá údržba. Opět jsem vytvořil metodu se vstupními parametry containterName pro název kontejneru a maintananceFarmName pro název Site, která se stará o údržbu. Po vytvoření hlavičky jsem ošetřil vstupy a následně jsem vytvořil funkci, která vrací seznam všech pravidel pro daný kontejner. Následně v cyklu jsem procházel daný seznam a pro každé pravidlo jsem testoval, zda už není v UrlRewrite pravidlech pro údržbu. Pokud nebylo, pak jsem jej tam vložil pomocí funkce, kterou jsem měl vytvořenou z prvního úkolu.

```
Function PlaceOnMaintenance {
        Param (
          [ string ]$kontejnerName=$(throw "Please␣specify␣Name␣of␣Kontejner."),
          [ string ]$MaintenanceFarmName=$maintenanceFarmConst
    )
    $log="‘nInserting␣container␣$($kontejnerName)␣into␣maintenance...."
    Out $log
    LoggingToFile -log $log
    $conditions=GetUrlRewriteConditionsForKontejner -kontainer $kontejnerName
     if ($conditions.count -gt 0)
    {
        ForEach($condition in $conditions)
        {
            ForEach($url in $condition.conditions.add.pattern)
            {
                if (-not (TestUrlRewriteCondition -wAppName $MaintenanceFarmName -url $url)
                    )
                {
                    AddUrlToUrlRewrite -wAppName $MaintenanceFarmName -url $url -
                        skipTestCondition
                }
            }
        }
        $log="‘nContainer␣$($kontejnerName)␣was␣successfully␣inserted␣into␣maintenance.‘n"
        LoggingToFile -log $log
        Write-Host $log -ForegroundColor Green
    }
    else
    {   $log="‘nNo␣items␣were␣found␣for␣container␣$($kontejnerName).Wrong␣name␣for␣
         container?"
        LoggingToFile -log $log
```

```
            Write-Host $log -ForegroundColor Red

        }
    }
```

Výpis 4: Ukázka metody pro vložení kontejneru do údržby

5.6.3 Logování událostí do souboru

Třetím úkolem bylo vytvoření funkce pro logování, kde vstupním parametrem byl řetězec, který se vkládal do logovacího souboru a následné začlenění funkce do již existujících funkcí. Požadavek pro logování byl z důvodu toho, že proces má být automatizovaný a případné hlášky a chyby by byly vypsány pouze na consoli, a poté by byly ztraceny.

6 Práce na firemních projektech

Po skončení školení a po domluvě s Ing. Romanem Hrdým a Ing. Radkem Petříkem jsme někteří byli vybráni a byla nám nabídnuta spolupráce na reálných projektech firmy. Každý z nás dostal vlastní pracovní místo, počítač a byly nám vytvořeny účty v rámci firmy, které se používaly pro přihlášení k počítači, e-mailu, komunikátoru Lync a aplikaci HelpDesk.

V této části budu popisovat úkoly, které se již plnily na reálných projektech, na kterých pracují zaměstnanci firmy. Všechny přidělované úkoly jsem dostával přes webovou aplikaci HelpDesk, ve které jsem byl k daným úkolům přiřazen. U každého úkolu bylo zadání, které stačilo, abych daný úkol pochopil a následně vypracoval. Pokud bylo zadání složitější, doptával jsem se na detaily přímo zadavatele úkolu ústně, nebo pomocí komunikátoru.

6.1 Testování aplikace xforlink

Prvním testovacím úkolem bylo testování webové aplikace xforlink a důkladný popis případných chyb pro vývojové oddělení. Nejprve jsem dostal mnohostránkový dokument, ve kterém byl popsán testovací scénář, podle kterého jsem měl postupovat. Po rychlém přečtení jsem se přihlásil do aplikace a začal jsem testovat krok po kroku podle scénáře. Když jsem narazil na chybu, vyzkoušel jsem postup ještě několikrát, jestli chyba není náhodná a poté jsem chybu i výskyt popsal do dokumentu.

Jakmile jsem prošel celým scénářem, začal jsem se seznamovat s ostatními moduly, které nebyly ve scénáři, a následně jsem je také otestoval. Tady už jsem měl volnou ruku, takže jsem testoval všechny možné situace, které mohou nastat. Občas jsem skutečně našel chybu, která nebyla jenom vzhledová, ale způsobila pád aplikace nebo její nekontrolovatelné chování, což může být horší, protože takové chyby si uživatel nemusí všimnout a může mít fatální následky. V tomto úkolu pro mě byla výhodou počáteční neznalost aplikace, protože při neznalosti správnosti postupu při práci s aplikací jsem mohl objevit chyby, které mohly být pro lidi, kteří s touto aplikací pracují, lehce přehlédnutelné.

6.2 Testování e-shopu

Druhým testovacím úkolem bylo testování e-shopu. Nejprve jsem dostal od vývojáře e-shopu adresu, na které e-shop běžel, a poté jsem mohl začít testovat. Testování jsem začal samotnou registrací na e-shopu, poté jsem pokračoval v testování změn kontaktních a registračních údajů. Dále jsem testoval zobrazování, filtrace a řazení zboží na stránce a nakonec jsem testoval práci s košíkem a vytvářením objednávek. Každou změnu, kterou jsem provedl v e-shopu od registrace až po vytváření objednávky, jsem mohl okamžitě vidět v aplikaci xforlink, protože tato aplikace byla pro e-shop administrativním rozhraním.

6.3 Tvorba testovacího scénáře pro xforlink

Zadáním tohoto úkolu bylo vytvoření testovacího scénáře pro různé části modulů, které ještě nebyly zdokumentovány. Abych mohl začít tvořit testovací scénář, musel jsem znát, jak se to vlastně dělá. Jelikož jsem nikdy nic podobného nevytvářel a bylo důležité, abych nezačal tvořit scénář jiným způsobem, než je ve firmě zaveden, podíval jsem se na již existující scénář, ve kterém bylo mnoho modulů popsáno. Následně jsem mohl začít v samotném psaní scénáře. Postupoval jsem tak, že jsem si nejprve prošel daný modul a seznamoval se s ním. Když jsem nevěděl, nebo jsem si nebyl jistý, jak daný proces probíhá, neváhal jsem se zeptat vývojářů aplikace, kteří mi vždy ochotně poradili.

Poté, když už jsem věděl, co se jak dělá a kde se mají data zobrazit, případně vložit, začal jsem popisovat dané kroky do dokumentu. Po napsání jsem si ještě jednou prošel celý postup, abych se ujistil že nic nechybí. Takto jsem pokračoval i ve zbylých modulech.

6.4 Tvorba číselníku „Typ přepravy“

Zadáním pro tento úkol bylo vytvořit nový číselník v sekci Obchod. Nejprve mi vedoucí vysvětlil, co obnáší tvorba takového jednoduchého číselníku, a jaké úkony musí být provedeny při tvorbě. Bylo potřeba vytvořit tabulku, k ní tři procedury a poté XML soubor, který popisoval zobrazení číselníku v programu.

Nejprve jsem musel v databázi vytvořit tabulku se sloupci Nazev datového typu varchar, Id, Jazyk a Vyrazen, všechny datového typu int. Poté bylo potřeba zajistit manipulaci s daty pro danou tabulku. Jelikož v databázi pro přístup k tabulkám se používají výhradně procedury, tak i já jsem musel vytvořit procedury, které zajistí všechny CRUD operace s tabulkou. Bylo potřeba vytvořit tři procedury.

První procedura slouží pro vložení a editaci záznamů. Na začátku procedury bylo nutné vytvořit kontrolu vstupních parametrů procedury, které byly nezbytné, aby bylo možné záznam vložit či upravit. Poté jsem vytvořil blok pro vložení záznamu a blok pro editaci záznamu. Pro zjištění, zda se bude jednat o vložení, či editaci záznamu, stačilo v podmínce kontrolovat, zda byl vložen do procedury primární klíč jako parametr, či nikoliv. Jestliže byl vložen primární klíč, pak se vstoupí do bloku pro editaci záznamu. Jestliže primární klíč nebyl vložen jako parametr, pak se provede získání prvního volného čísla a provede se vložení záznamu do tabulky.

Druhá procedura slouží pro zobrazení záznamů. Vytvořil jsem jediný dotaz, pro který jsem v podmínce selekce záznamů přidal vstupní parametry procedury, podle kterých se můžou záznamy filtrovat z mnou vytvořené tabulky.

Třetí procedura slouží pro mazání záznamů. Jelikož se v databázi žádné záznamy nemažou, je mazání řešeno editací mazaného záznamu, a to tak, že se hodnota ve sloupci Vyrazen nastaví na 1. Tím se záznam nevymaže, ale je uchován pro případnou obnovu.

Když už byla databázová část hotová, bylo potřeba přidat číselník do programu. Jelikož se jednalo o číselník, přidání do programu nebylo nijak složité. Nejprve bylo potřeba přidat záznam do souboru, který popisuje, jak je daný číselník zobrazován v programu a v jakém pořadí budou sloupce tabulky zobrazovány. Případně bylo také zapotřebí přidat popis sloupce, který bude zobrazen v programu a nakonec které sloupce budou viditelné a bude je možno editovat.

Poté, když už bylo definováno, jak bude číselník zobrazen v programu, bylo nutné specifikovat, které procedury budou s číselníkem pracovat, když bude potřeba data načíst, editovat či mazat. Řešením bylo přidání nového záznamu do XML souboru obsahující jména procedur pro práci s tabulkou a dále popis vstupních parametrů pro každou z procedur.

6.5 Zrušení nepoužívaných číselníků

Zadáním pro tento úkol bylo zrušení všech nepoužívaných číselníků z modulu Obchod, které byly v aplikaci nadbytečné. Nejprve jsem si spustil aplikaci a našel jsem si číselník, který jsem měl za úkol zrušit. Poté jsem si v kódu našel, kde se daný číselník zobrazuje a načítá z databáze, a tento kód jsem odstranil. Nakonec jsem si našel XML soubor, který popisuje, jak vypadají dané sloupce pro číselník v aplikaci, a tento soubor odstranil. Tímto způsobem jsem odstranil všechny požadované číselníky a poté jsem změny nahrál na TFS. Jelikož v zadání bylo, že mám číselníky odebrat pouze z aplikace, procedury a samotné tabulky číselníků jsem v databázi nechal nezměněny.

6.6 Generování smlouvy

V tomto úkolu šlo o získání dat z databáze a poté o vyplnění příslušných dokumentů těmito daty. Pro tento účel jsem měl použít knihovnu OpenXml, která dokáže pracovat s Microsoft Office dokumenty. Jelikož jsem vůbec nevěděl nic o této knihovně a hlavně nikdo ve firmě nic podobného nevytvářel, nezbývalo mi nic jiného než prozkoumat tuto knihovnu sám.

V první fázi jsem si připravil šablonu smlouvy, a to tak, že jsem si na místa, kde se mají vkládat data, přidával mnou vytvořená klíčová slova. Zpočátku jsem knihovnu využíval pouze k otevření souboru a přečtení celého obsahu do řetězce, kde jsem pomocí metody nahrazoval mnou vytvořená klíčová slova za data. Vše vypadalo jednoduše a pohodlně až do doby, kdy jsem musel začít přidávat názvy příloh do tabulky ve spodní části dokumentu. Zde jsem již nemohl použít obyčejné nahrazení slov. Díky tomu jsem musel vytvořit metodu, která dokáže nalézt tabulku v dokumentu a poté do této tabulky vkládat nové řádky a sloupce podle počtu příloh. Metoda pro přidávání nových buněk pro řádek obsahovala tři parametry. Prvním parametrem byl odkaz na řádek tabulky, druhým text, který se má přidat do buňky a nakonec velikost písma pro danou buňku. Na ukázkovém kódu 5 lze vidět, že se řádek tabulky skládá z buněk. Každá buňka má v

sobě paragraf, který se skládá z kolekce instancí Run třídy. Instance Run třídy dokáže v sobě držet text a formát pro daný text. Díky tomu lze v dokumentu vytvořit např. slovo, které může mít část napsanou tučně a část kurzívou. Toto slovo se bude skládat ze dvou instancí třídy Run.

```
private static void fillRowCell(TableRow tr,string cellText, int size)
{
    TableCell addrCell = tr.AppendChild(new TableCell());
    Paragraph para = addrCell.AppendChild(new Paragraph());
    Run run = para.AppendChild(new Run());
    RunProperties runProperties = run.AppendChild(new RunProperties(new RunFonts() { Ascii = "
        Arial", HighAnsi = "Arial" }));
    FontSize fontSize = new FontSize() { Val = size.ToString() };
    runProperties.AppendChild(fontSize);

    run.AppendChild(new Text(cellTextList[i]));
}
```

Výpis 5: Metoda pro přidání nové buňky pro řádek tabulky

Po dokončení šablony pro smlouvu a všech metod pro nahrazení textu v dokumentu jsem si vytvořil proceduru pro získání dat z databáze. Výsledná procedura obsahovala 8 spojení tabulek a vracela pouze jeden záznam se 17 sloupci. Poté jsem si zavedl volání procedury do programu a nakonec jsem provedl nahrazení klíčových slov ostrými daty. V další části jsem musel vytvořit formulář, ve kterém uživatel zadá požadovaná vstupní data nutná k tomu, aby bylo možné smlouvu vygenerovat.

Poté jsem dostal 12 příloh a 12 dodatků, které vždy obsahovaly tabulku a data zákazníka. Pro každou přílohu jsem musel vytvořit šablonu, kde jsem opět vkládal vlastní klíčová slova, z nichž jedno nahrazovalo vygenerovanou tabulku. Nakonec jsem musel vytvořit ještě smlouvu ve druhém jazyce a k ní dva dodatky obdobným způsobem, který jsem již popisoval.

6.7 Úprava aplikace Imagedigger, přídání logování změněných souborů

Mým úkolem bylo v již existující aplikaci Imagedigger, která slouží pro změnu velikosti obrázku, vyhledat místo, kde se provádí změna velikosti obrázků, a pokud u daného obrázku dojde ke změně, pak je potřeba zaznamenat název obrázku do logu, který bude umístěn ve zdrojové rootovské složce. Implementace samotného logování byla jednoduchá, horší bylo se zorientovat v cizím kódu a nalézt ono místo, kde se prováděla změna velikosti.

7 Znalosti a dovednosti

7.1 Získané odborné znalostí a dovednosti ze studia a jejich využití

Díky znalostem získaným ze studia na VŠB-TU v Ostravě v oblasti programování a databází jsem byl dostatečně připraven na přijímací pohovor, kde jsem musel podstoupit vstupní test, který byl rozdělen na tři části. První část byla teoretická a týkala se programovacího jazyka C#. V druhé části jsem musel pro požadované zadání nakreslit, jak by měly vypadat tabulky a vazby mezi nimi. Dále v průběhu praxe jsem maximálně využíval znalosti programovacího jazyka C# a MS SQL, které byly nezbytnou součástí pro plnění požadovaných úkolů během praxe.

7.2 Chybějící odborné znalosti a dovednosti

V průběhu praxe jsem musel čelit mnoha neznámým věcem, které jsem doposud neznal. Byly to např. webové služby, služby ve Windows, vývoj aplikací v ASP.NET, práce s logováním událostí v operačním systému Windows a výměna dat mezi aplikacemi. Nicméně mě neznalost těchto záležitostí neodradila, nýbrž naopak, s chutí jsem se učil novým věcem a prozkoumával neznámé oblasti, které bylo nutné znát pro to, abych dokázal řešit dané úkoly.

8 Závěr

V průběhu praxe jsem plnil mnoho různých úkolů. Často jsem pracoval s novými technologiemi, díky čemuž jsem získal mnoho cenných zkušeností, které se dají získat pouze v praxi. Pouze zde na praxi jsem si mohl vyzkoušet, co je to pracovat na reálných projektech vyvíjených pro zákazníky podle jejich požadavků a také jsem si vyzkoušel práci v týmu. Úkoly, které mi byly zadány, jsem se snažil plnit co nejsvědomitěji a až do konce, což se mi i dle mého názoru dařilo. Jsem rád, že jsem mohl vykonávat odbornou praxi ve firmě a měl jsem možnost vidět, jak firma funguje. Také jsem viděl metodiku agilního programování s názvem Scrum. Všichni spolupracovníci byli přátelští a ochotní mi s čímkoliv vyjít vstříc a vysvětlit mi případný problém, když jsem si nevěděl rady. Díky tomu jsem měl pocit, že mě berou jako svého spolupracovníka a ne pouze jako nějakého studenta, který přišel vykonat praxi.

Praxe pro mne byla velikým přínosem jak z hlediska nově nabytých znalostí a zkušeností, ale také z hlediska zodpovědnosti a pečlivosti při plnění zadaných úkolů. Díky této praxi jsem mohl během delšího období pozorovat, jak vypadá chod firmy a sám jsem si to i vyzkoušel. Na závěr bych chtěl ještě jednou poděkovat všem, kteří mi pomáhali jak v počátcích, tak v průběhu celé praxe.

9 Reference

[1] E LINKX a. s., *O společnosti* [online], [cit. 2015-03-21]. Dostupné z: (http://www.elinkx.cz/o-spolecnosti)

[2] E LINKX a. s., *Jeden e-shop pro všechny - xShop.link* [online], [cit. 2015-03-21]. Dostupné z: (http://www.elinkx.cz/xshop)

[3] E LINKX a. s., *xSklad - sklad s přehledem* [online], [cit. 2015-03-21]. Dostupné z: (http://www.elinkx.cz/xsklad)

[4] E LINKX a. s., *Informační systém Esyco.NET* [online], [cit. 2015-03-21]. Dostupné z: (http://www.elinkx.cz/esyco-net)

[5] E LINKX a. s., *Technická podpora HelpDesk* [online], [cit. 2015-03-21]. Dostupné z: (http://www.elinkx.cz/helpdesk)

[6] WinMerge Development Team, *WinMerge* [online] [cit. 2015-03-21]. Dostupné z: (http://winmerge.org/about/)

[7] Microsoft®, *Team Foundation Server* [online], [cit. 2015-03-21]. Dostupné z: (https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx)

[8] Microsoft®, *Microsoft® SQL Server® 2008 Management Studio* [online], [cit. 2015-03-21]. Dostupné z: (https://www.microsoft.com/en-us/download/details.aspx?id=7593)

[9] Microsoft®, *Visual C#* [online], [cit. 2015-03-21]. Dostupné z: (https://msdn.microsoft.com/cs-cz/library/kx37x362.aspx)

[10] Microsoft®, *Get Started with ASP.NET* [online], [cit. 2015-03-21]. Dostupné z: (http://www.asp.net/get-started)

[11] Microsoft®, *Windows PowerShell Scripting* [online], [cit. 2015-03-21]. Dostupné z: (https://technet.microsoft.com/en-us/scriptcenter/powershell.aspx)

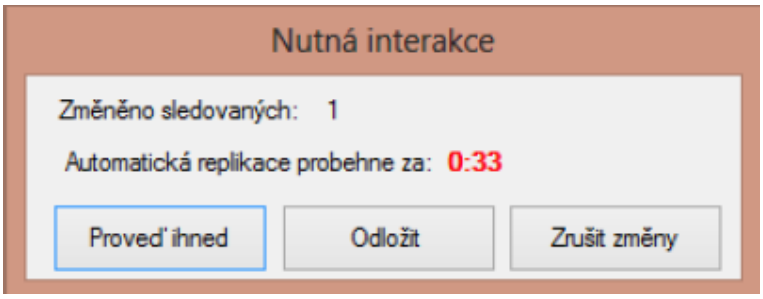
[12] Microsoft®, *Web Server (IIS) Administration Cmdlets in Windows PowerShell* [online], [cit. 2015-03-21]. Dostupné z: (https://technet.microsoft.com/en-us/library/ee790599.aspx)

[13] Microsoft®, *Word processing (Open XML SDK)* [online]. Poslední změna 27.07.2012, [cit. 2015-03-21]. Dostupné z: (https://msdn.microsoft.com/en-us/library/office/cc850833.aspx)

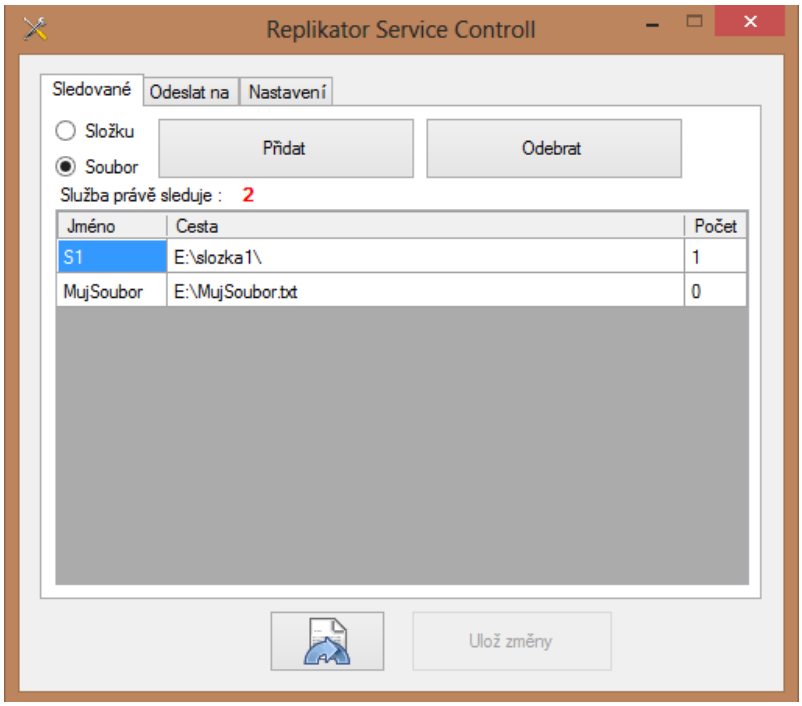
[14] Vithal Wadje, *Consuming Web Service In an ASP.Net Web Application* [online]. Datum publikování 02.06.2013, [cit. 2015-03-21]. Dostupné z: (http://www.c-sharpcorner.com/UploadFile/0c1bb2/consuming-web-service-in-Asp-Net-web-application/)

[15] Microsoft®, *Walkthrough: Creating a Windows Service Application in the Component Designer* [online], [cit. 2015-03-21]. Dostupné z: (https://msdn.microsoft.com/en-us/library/zt39148a(v=vs.110).aspx)

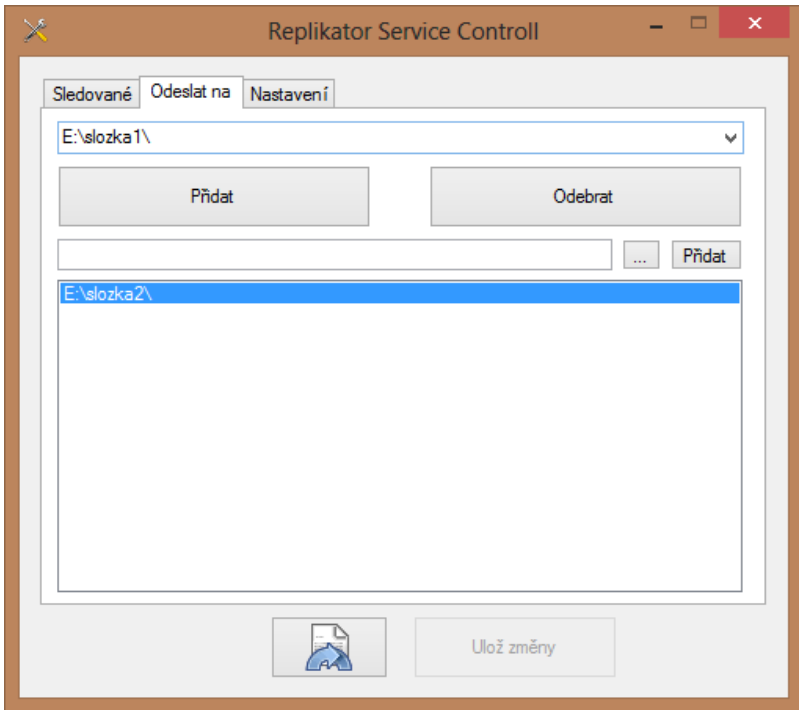
A Replikator



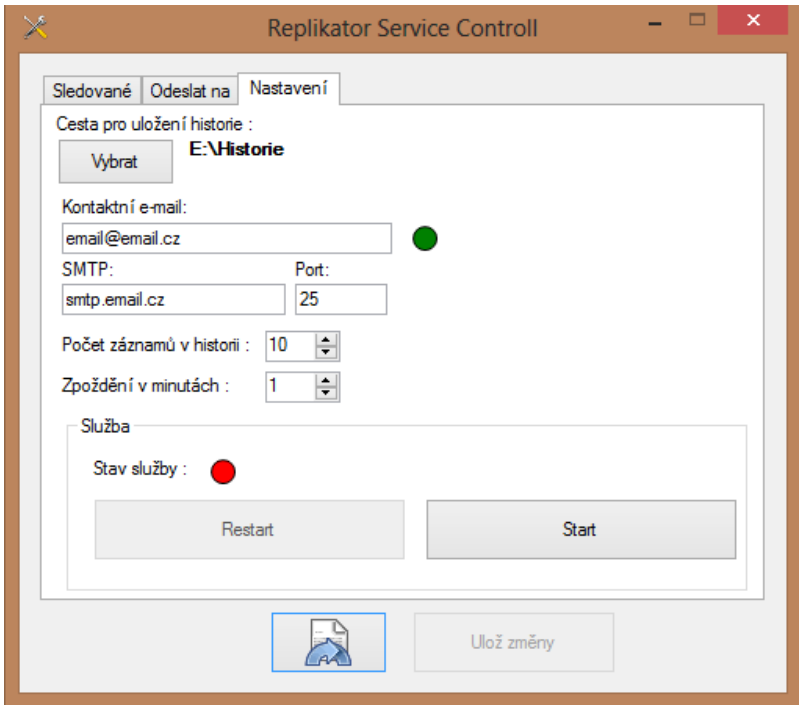
Obrázek 2: Ukázka okna pro práci se změnami



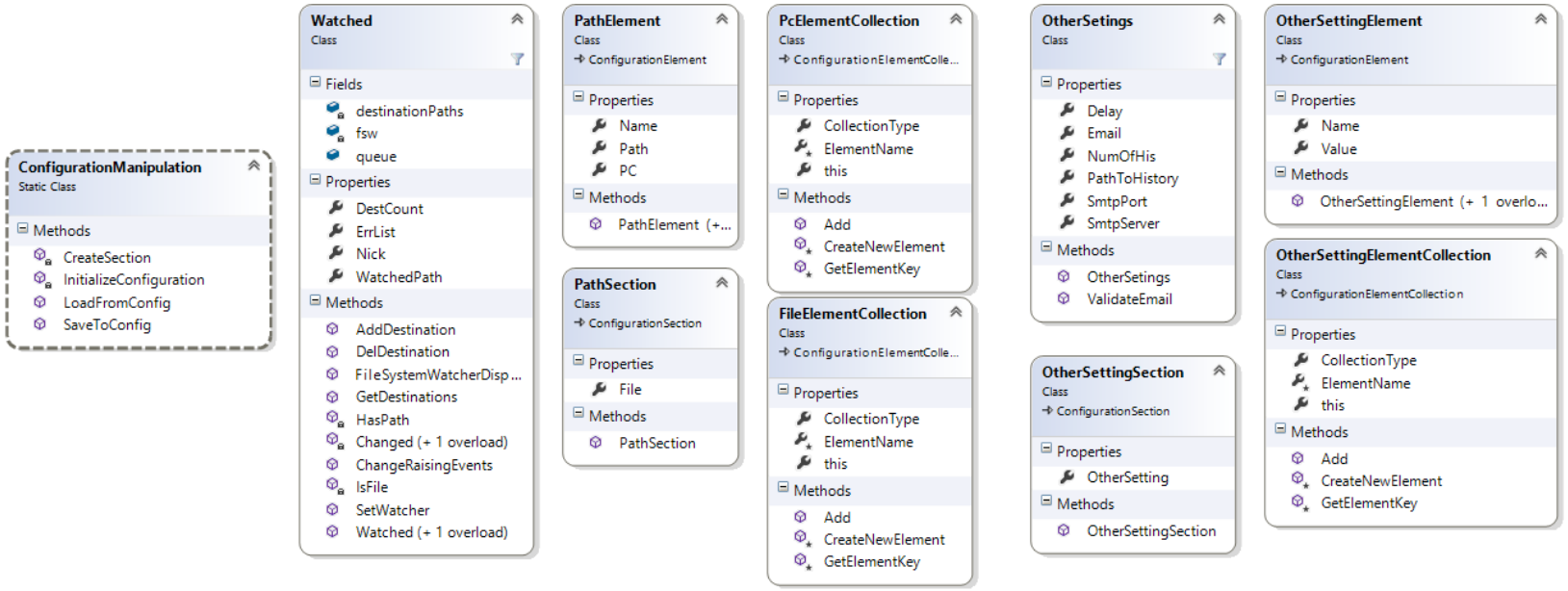
Obrázek 3: Ukázka okna pro nastavení sledovaných souborů či složek



Obrázek 4: Ukázka okna pro namapování sledovaných na cílové cesty



Obrázek 5: Ukázka okna pro nastavení a manipulaci se službou



Obrázek 6: Třídní diagram knihovny tříd pro práci s konfiguračním souborem

B Seznam všech úkolů a jejich časová náročnost

Název úkolu	**Hodiny [h]**
Tvorba webové aplikace	61
Tvorba konfigurační aplikace	18
Práce na službě	40
Práce na notifikační aplikaci	44
Tvorba technické dokumentace	19
Tvorba nasazovacího skriptu	44
Feedy	3
Analýza problému s xformax aplikací	4
Tvorba XML transformací	24
Testování xforlink aplikace	9,4
Testování e-shopu	7
Tvorba číselníků	26,5
Zrušení nepoužívaných číselníků	1,5
Obchodní modul - tvorba nových modulů	174
Obchodní modul - generování smlouvy	139
Plán práce pro OM - oprava chyby v aplikaci	1,3
Změna práv v reportech výkonů	7,3
Úprava šířek sloupců pro GridView v celé aplikaci	16
Nahrazení původního control prvku za nový prvek v celé aplikaci	19
Oprava chyb aplikace	40
Modifikace stávajících funkcí aplikace	35
Přidání nových dat do zprávy o škodě	7

Tabulka 1: Seznam úkolů a jejich časová náročnost